ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

Software for Nonlinearly Constrained Optimization

Sven Leyffer and Ashutosh Mahajan

Mathematics and Computer Science Division

Preprint ANL/MCS-P1768-0610

June 17, 2010

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

Contents

Software for Nonlinearly Constrained Optimization*

Sven Leyffer† and Ashutosh Mahajan ‡

June 17, 2010

Abstract

We categorize and survey software packages for solving constrained nonlinear optimization problems, including interior-point methods, sequential linear/quadratic programming methods, and augmented Lagrangian methods. In each case we highlight the main methodological components and provide a brief summary of interfaces and availability.

Keywords: Mathematical programming methods, Newton-type methods, nonlinear programming, interior-point methods, sequential quadratic programming, sequential linear programming

AMS-MSC2000: 49M05, 49M15, 49M37, 65K05, 90C30, 90C51, 90C55

1 Introduction and Background

Software for nonlinearly constrained optimization (NCO) tackles problems of the form

$$\begin{cases} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & l_c \le c(x) \le u_c \\ & l_A \le A^T x \le u_A \\ & l_x \le x \le u_x, \end{cases} \tag{1.1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ and the constraint functions $c_i : \mathbb{R}^n \to \mathbb{R}$, for $i = 1, \ldots, m$ are twice continuously differentiable. The bounds $l_c, l_A, l_x, u_c, u_A, u_x$ can be either finite or infinite. Equality constraints are modeled by setting $l_j = u_j$ for some index j. Maximization problems can be solved by multiplying the objective by -1 (most solvers handle this transformation internally). Solvers often take advantage of special constraints such as simple bounds and linear constraints.

NCO solvers are typically designed to work well for a range of other optimization problems such as solving a system of nonlinear equations (most methods reduce to Newton's method in

*Preprint ANL/MCS-P1768-0610.

†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, leyffer@mcs.anl.gov.

‡Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, mahajan@mcs.anl.gov.

this case), bound-constrained problems, and LP or QP problems. In this survey, we concentrate on solvers that can handle general NCO problems possibly involving nonconvex functions. There is software like, for instance, MOSEK (Andersen et al., 2009) and CVXOPT (Dahl and Vandenberghe, 2010), that is designed specifically for problems with convex functions only.

Methods for solving (1.1) are iterative. That is, given an estimate x_0 of the solution, they compute a sequence of iterates $\{x_k\}$ converging to a stationary point, using the following four basic components:

1. **Approximate Subproblem.** The subproblem is an approximation of (1.1) around the current iterate x_k that can be solved (approximately), and produces a step or search direction that improves the current estimate x_k. Examples of such subproblems are linear or quadratic programming approximation, the barrier problem, and an augmented Lagrangian; see Chapter **??**.

2. **Global Convergence Strategy.** The (local) subproblem alone cannot guarantee convergence to a stationary point, and optimization methods need a convergence strategy that forces the sequence $\{x_k\}$ to converge. Examples of convergence strategies are penalty functions, augmented Lagrangian, filter, and funnel methods; see Chapter **??**.

3. **Global Convergence Mechanism.** The global convergence strategy is enforced by using a mechanism that improves the subproblem by shortening the step that is computed. There are two classes of convergence mechanism: line search and trust region.

4. **Convergence Test.** A convergence test is needed to stop the algorithm once the error in the Karush-Kuhn-Tucker (or Fritz-John) conditions is below a user-specified tolerance. In addition, solvers may converge only to a local minimum of the constraint violation.

Solvers for NCO are differentiated by how each of these key ingredients is implemented. In addition there are a number of secondary distinguishing factors such as licensing (open-source versus commercial or academic), API and interfaces to modeling languages, sparse or dense linear algebra, programming language (Fortran/C/MATLAB), and compute platforms on which a solver can run.

The next sections list some solvers for NCOs, summarizing their main characteristics. A short overview can be found in Table 1. We distinguish solvers by the definition of their subproblem. An alternative characterization could be based on the way in which solvers handle constraints.

2 Interior-Point Solvers

Interior-point methods approximately solve a sequence of perturbed KKT systems, driving a barrier parameter to zero. Interior-point methods can be regarded as perturbed Newton methods applied to the KKT system in which the primal/dual variables are kept positive.

Table 1: NCO Software Overview.

Name	Model	Global Method	Interfaces	Language
ALGENCAN	Aug. Lag.	penalty	AMPL, C/C++, CUTEr, Java, Matlab, Octave, Python, R	f77
CONOPT	GRG/SLQP	line search	AIMMS, GAMS	fortran
FilterSQP	SQP	filter/trust region	AMPL, CUTEr, f77	fortran 77
GALAHAD	Aug. Lag.	nonmonotone/ aug. Lagrangian	CUTEr, fortran	Fortran95
Ipfilter	IPM	filter/line search	AMPL, CUTEr, f90	Fortran 90
IPOPT	IPM	filter/line search	AMPL, CUTEr, C, C++, f77	C++
KNITRO	IPM	penalty/trust region line-search	AIMMS, AMPL, GAMS, Mathematica, MATLAB, MPL, C, C++, f77, Java, Excel	C
KNITRO	SLQP	penalty/trust region	s.a.	C
LANCELOT	Aug. Lag.	augmented Lagrangian/ trust region	SIF, AMPL, f77	f77
LINDO	GRG/SLP	only convex	C, MATLAB, LINGO	
LOQO	IPM	line search	AMPL, C, MATLAB	C
LRAMBO	SQP	ℓ_1 exact penalty/ line search	C	C/C++
NLPQLP	SQP	augmented Lagrangian/ line search	C, f77, MATLAB	f77
MINOS	Aug. Lag.	None	AIMMS, AMPL, GAMS, MATLAB, C, C++, f77	f77
PATH	LCP	line search	AMPL	C
PENNON	Aug. Lag.	line search	AMPL, MATLAB	C
NPSOL	SQP	penalty Lagrangian/ line search	AIMMS, AMPL, GAMS, MATLAB, C, C++, f77	f77
SNOPT	SQP	penalty Lagrangian/ line search	AIMMS, AMPL, GAMS, MATLAB, C, C++, f77	f77
SQPlab	SQP	penalty Lagrangian/ line search	MATLAB	MATLAB

2.1 Ipfilter

Algorithmic Methodology: Ipfilter is based on the interior point filter method of Ulbrich et al. (2004). A step is first generated by solving the normally used KKT system. It then uses a specially

designed filter for selecting the step size in the direction of the calculated step. The first component of the filter is the sum of "constraint violation" and "centrality", while the second component is the norm of the gradient of the Lagrangean. This method can sometimes converge to a local maximum because the filter does not explicitly check the objective function value. In the implementation of Ipfilter, the second component of the filter is modified to overcome this problem (Ulbrich et al., 2008).

Software and Technical Details: Ipfilter version 0.3 is written in Fortran90. It requires a user to have MA57 routine from the HSL library. Ipfilter is available at no cost for academic purposes and can be obtained by contacting the developers at the Ipfilter webpage (Silva et al., 2010). It has interfaces in AMPL and CUTEr. The user can also write their own functions and derivatives in Fortran90.

2.2 IPOPT

Algorithmic Methodology: IPOPT is a line-search filter interior-point method (Wächter and Biegler, 2005b,a; Wächter and Biegler, 2006). The outer loop approximately minimizes a sequence of nonlinearly (equality) constrained barrier problems for a decreasing sequence of barrier parameters. The inner loop uses a line-search filter SQP method to approximately solve each barrier problem. Global convergence of each barrier problem is enforced through a line-search filter method, and the filter is reset after each barrier parameter update. Steps are computed by solving a primal-dual system, (**??**), corresponding to the KKT conditions of the barrier problem. The algorithm controls the inertia of this system by adding δI ($\delta > 0$) to the Hessian of the Lagrangian, ensuring descent properties. The inner iteration includes second-order correction steps and mechanisms for switching to a feasibility restoration if the step size becomes too small. The solver has an option for using limited-memory BFGS updates to approximate the Hessian of the Lagrangian.

Software and Technical Details: The solver is written in C++ and has interfaces to C, C++, Fortran, AMPL, CUTEr, and COIN-OR's NLPAPI. It requires BLAS, LAPACK, and a sparse indefinite solver (MA27, MA57, PARDISO, or WSMP). The user (or the modelling language) must provide function and gradient information, and possibly the Hessian of the Lagrangian (in sparse format). By using the PARDISO (Schenk et al., 2007) parallel linear solver, IPOPT can solve large problems on shared-memory multiprocessors. IPOPT is available on COIN-OR under Common Public License at its website (Wächter and Biegler, 2010).

2.3 KNITRO

KNITRO includes both IPM and SLQP methods. The IPM version is described in this section.

Algorithmic Methodology: KNITRO implements both trust-region based and line-search based interior-point/barrier methods (Byrd et al., 1999, 2000, 2006). It approximately solves a sequence

of barrier subproblems for a decreasing sequence of barrier parameters. It uses a merit or a penalty function for accepting a step and for ensuring global convergence. The barrier subproblems are solved by a sequence of linearized primal-dual equations, (**??**). KNITRO has two options for solving the primal-dual system: direct factorization of the system of equations and preconditioned conjugate gradient (PCG) method. The PCG method solves the indefinite primal-dual system by projecting onto the null space of the equality constraints. KNITRO also includes modules for solving mixed-integer nonlinear optimization problems and optimization problems with simple complementarity constraints. In addition, it contains crossover techniques to obtain an active set from the solution of the IPM solve and a multistart heuristic for nonconvex NCOs.

Software and Technical Details: KNITRO is written in C. It has interfaces to a range of modeling languages, including AIMMS, AMPL, GAMS, Mathematica, MATLAB, and MPL. In addition, KNITRO has interfaces to C, C++, Fortran, Java, and Excel. It offers callback and reverse communication interfaces. KNITRO makes use of MA27/MA57 to solve the indefinite linear systems of equations, and these routines are included in the package. KNITRO is available from Ziena Optimization, Inc., and the user's manual is available online (Waltz and Plantenga, 2009).

2.4 LOQO

Algorithmic Methodology: LOQO (Vanderbei and Shanno, 1999) uses an infeasible primal-dual interior-point method for solving general nonlinear problems. The inequality constraints are added to the objective by using a log-barrier function. Newton's method is used to obtain a solution to the system of nonlinear equations that is obtained by applying first-order necessary conditions to this barrier function. The solution of this system provides a search direction, and a filter or a merit function is used to accept the next iterate obtained by performing a line search in this direction. The filter is specially modified for interior point methods (Benson et al., 2002). It allows a point that improves either the feasibility or the barrier objective as compared to the current iterate only. The merit function is the barrier function and an additional ℓ_2 norm of the violation of constraints. The exact Hessian of the Lagrangian is used in the Newton's method. When the problem is nonconvex, this Hessian is perturbed by adding to it the matrix δI, where I is an identity matrix and $\delta > 0$ is chosen so that the perturbed Hessian is positive definite. This perturbation ensures that if the iterates converge, they converge to a local minimum. LOQO can also handle complementarity constraints by using a penalty function (Benson et al., 2006).

Software and Technical Details: LOQO can be used to solve problems written in AMPL. The user can also implement functions in C and link to the LOQO library. LOQO can also read files in MPS format for solving LPs. A MATLAB interface for LOQO is available for LPs, QPs, and second-order cone programs. The library and binary files are available for a license fee at its homepage (Vanderbei, 2010a), while a limited student version is available freely. A user manual provides instructions on installing and using LOQO.

3 Sequential Linear/Quadratic Solvers

Sequential linear/quadratic solvers solve a sequence of LP and/or QP subproblems. This class of solvers is also referred to as active-set methods, because they provide an estimate of the active set at every iteration. Once the active set settles down, these methods become Newton methods on the active constraints (except SLP).

3.1 CONOPT

Algorithmic Methodology: CONOPT (Drud, 1985, 2007) implements three active-set methods. The first is a gradient projection method that projects the gradient of the objective onto a linearization of the constraints and makes progress toward the solution by reducing the objective. The second variant is an SLP method, and the third is an SQP method. CONOPT includes algorithmic switches that automatically detect which method is preferable. It also exploits triangular parts of the Jacobian matrix and linear components of the Jacobian to reduce the sparse factorization overhead. CONOPT is a line-search method.

Software and Technical Details: CONOPT has interfaces to AMPL, GAMS and AIMMS and is available from any of the three companies.

3.2 FilterSQP

Algorithmic Methodology: FilterSQP (Fletcher and Leyffer, 1998) implements a trust-region SQP method. Convergence is enforced with a filter, whose components are the ℓ_1-norm of the constraint violation, and the objective function. The solver starts by projecting the user-supplied initial guess onto the linear part of the feasible set and remains feasible with respect to the linear constraints. It uses an indefinite QP solver that finds local solutions to problems with negative curvature. The solver switches to a feasibility restoration phase if the local QP model becomes inconsistent. During the restoration phase, a weighted sum of the constraint violations is minimized by using a filter SQP trust-region approach. The restoration phase either terminates at a local minimum of the constraint violation or returns to the main algorithm when a feasible local QP model is found. FilterSQP computes second-order correction steps if the QP step is rejected by the filter.

Software and Technical Details: The solver is implemented in Fortran77 and has interfaces to AMPL, CUTEr, and Fortran77. The code requires subroutines to evaluate the problem functions, their gradients, and the Hessian of the Lagrangian (provided by AMPL and CUTEr). It uses BQPD (Fletcher, 1999) to solve the possibly indefinite QP subproblems. BQPD is a null-space active-set method and has modules for sparse and dense linear algebra with efficient and stable factorization updates. The code is licensed by the University of Dundee.

3.3 KNITRO

Algorithmic Methodology: In addition to the interior point-methods (see Section 2.3), KNITRO implements an SLQP algorithm (Fletcher and de la Maza, 1989; Byrd et al., 2004). In each iteration, an LP that approximates the ℓ_1 exact-penalty problem is solved to determine an estimate of the active set. This LP also has an additional infinity-norm trust-region constraint. Those constraints in the LP that are satisfied as equalities are marked as active and are used to set up an equality-constrained QP (EQP), (**??**), whose objective is a quadratic approximation of the Lagrangian of (1.1) at the current iterate. An ℓ_2-norm trust-region constraint is also added to this QP. The solution of the LP and the EQP are then used to find a search direction (Byrd et al., 2006, 2004). A projected conjugate-gradient method is used to solve the EQP. The penalty parameter ρ_k is updated to ensure sufficient decrease toward feasibility.

Software and Technical Details: The SLQP method of KNITRO requires an LP solver. KNITRO includes CLP as well as a link to CPLEX as options. For other details, see Section 2.3. This active-set algorithm is usually preferable for rapidly detecting infeasible problems and for solving a sequence of closely related problems.

3.4 LINDO

Algorithmic Methodology: LINDO provides solvers for a variety of problems including NCOs. Its nonlinear solver implements an SLP algorithm and a generalized gradient method. It provides options to randomly select the starting point and can estimate derivatives by using finite differences.

Software and Technical Details: LINDO (LINDO Systems Inc., 2010a) provides interfaces for programs written in languages such as C and MATLAB. It can also read models written in LINGO. The full version of the software can be bought online, and a limited trial version is available for free (LINDO Systems Inc., 2010b).

3.5 LRAMBO

Algorithmic Methodology: LRAMBO is a *total quasi-Newton* SQP method. It approximates both the Jacobian and the Hessian by using rank-1 quasi-Newton updates. It enforces global convergence through a line search on an ℓ_1-exact penalty function. LRAMBO requires as input only an evaluation program for the objective and the constraints. It combines automatic differentiation and quasi-Newton updates to update factors of the Jacobian, and it computes updates of the Hessian. Details can be found in (Griewank et al., 2007).

Software and Technical Details: LRAMBO uses the NAG subroutine E04NAF to solve the QP subproblems. It also requires ADOL-C (Griewank et al., 1996; Griewank and Walther, 2004) for

the automatic differentiation of the problem functions. LRAMBO is written in C/C++.

3.6 NLPQLP

Algorithmic Methodology: NLPQLP is an extension of the SQP solver NLPQL (Schittkowski, 1985) that implements a nonmonotone line search to ensure global convergence. It uses a quasi-Newton approximation of the Hessian of the Lagrangian, which is updated with the BFGS formula. A nonmonotone line search is used to calculate the step length that minimizes an augmented Lagrangian merit function.

Software and Technical Details: NLPQLP is implemented in Fortran. The user or the interface is required to evaluate the function values of the objective and constraints. Derivatives, if unavailable, can be estimated by using finite differences. NLPQLP can evaluate these functions and also the merit function on a distributed memory system. A user guide with documentation, algorithm details, and examples is available (Schittkowski, 2009). NLPQLP can be obtained under academic and commercial license from its website.

3.7 NPSOL

Algorithmic Methodology: NPSOL (Gill et al., 1998) solves general nonlinear problems by using an SQP algorithm with a line search on the augmented Lagrangian. In each major iteration, a QP subproblem is solved whose Hessian is a quasi-Newton approximation of the Hessian of the Lagrangian using a *dense* BFGS update.

Software and Technical Details: NPSOL is implemented in Fortran, and the library can be called from Fortran and C programs and from modeling languages such as AMPL, GAMS, AIMMS, and MATLAB. The user or the interface must provide routines for evaluating functions and their gradients. If a gradient is not available, NPSOL can estimate it by using finite differences. It treats all matrices as dense and hence may not be efficient for large-sparse problems. NPSOL can be warm started by specifying the active constraints and multiplier estimates for the QP. NPSOL is available under commercial and academic licenses from Stanford Business Software Inc.

3.8 PATHNLP

Algorithmic Methodology: PATH (Dirkse and Ferris, 1995; Ferris and Munson, 1999) solves mixed complementarity problems (MCPs). PATHNLP automatically formulates the KKT conditions of an NLP, (1.1), specified in GAMS as an MCP and then solves this MCP using PATH. The authors note that this approach is guaranteed only to find stationary points and does not distinguish between local minimizers and maximizers for nonconvex problems. Thus, it works well for convex problems. At each iteration, PATH solves a linearization of the MCP problem to obtain a Newton point. It then performs a search in the direction of this point to find a minimizer of a

merit function. If this direction is not a descent direction, it performs a steepest descent step in the merit function to find a new point.

Software and Technical Details: The PATHNLP is available only through GAMS (Dirkse and Ferris, 2010b). The PATH solver is implemented in C and C++ (Dirkse and Ferris, 2010a).

3.9 SNOPT

Algorithmic Methodology: SNOPT (Gill et al., 2006a) implements an SQP algorithm much like NPSOL, but it is suitable for large, sparse problems as well. The Hessian of the Lagrangian is updated by using limited-memory quasi-Newton updates. SNOPT solves each QP using SQOPT (Gill et al., 2006b), which is a reduced-Hessian active-set method. It includes an option for using a projected conjugate gradient method rather than factoring the reduced Hessian. SNOPT starts by solving an "elastic program" that minimizes the constraint violation of the linear constraints of (1.1). The solution to this program is used as a starting point for the major iterations. If a QP subproblem is found to be infeasible or unbounded, then SNOPT tries to solve an elastic problem that corresponds to a smooth reformulation of the ℓ_1-exact penalty function The solution from a major iteration is used to obtain a search direction along which an augmented Lagrangian merit function is minimized.

Software and Technical Details: SNOPT is implemented in Fortran77 and is compatible with newer Fortran compilers. All functions in the SNOPT library can be used in parallel or by multiple threads. It can be called from other programs written in C and Fortran and by packages such as AMPL, GAMS, AIMMS, and MATLAB. The user or the interface has to provide routines that evaluate function values and gradients. When gradients are not available, SNOPT uses finite differences to estimate them. SNOPT can also save basis files that can be used to save the basis information to warm start subsequent QPs. SNOPT is available under commercial and academic licenses from Stanford Business Software Inc.

3.10 SQPlab

Algorithmic Methodology: SQPlab (Gilbert, 2009) was developed as a laboratory for testing algorithmic options of SQP methods (Bonnans et al., 2006). SQPlab implements a line-search SQP method that can use either exact Hessians or a BFGS approximation of the Hessian of the Lagrangian. It maintains positive definiteness of the Hessian approximation using the Wolfe or Powell condition. SQPlab uses a Lagrangian penalty function, $p_\sigma(x, y) = f(x) + y_c^T c(x) + \sigma \| \max\{0, c(x) - u_c, l_c - c(x)\} \|_1$, to promote global convergence. SQPlab has a feature that allows it to treat discretized optimal control constraints specially. The user can specify a set of equality constraints whose Jacobian has uniformly full rank (such as certain discretized optimal-control constraints). SQPlab then uses these constraints to eliminate the state variables. The user needs to provide only routines that multiply a vector by the inverse of the control constraints.

Software and Technical Details: SQPlab is written in MATLAB and requires quadprog.m from the optimization toolbox. The user must provide a function simulator to evaluate the functions and gradients. A smaller version is also available that does not require quadprog.m but instead uses qpal.m, an augmented Lagrangian QP solver for medium-sized convex QPs. All solvers are distributed under the Q public license from INRIA, France (Gilbert, 2010).

4 Augmented Lagrangian Solvers

Augmented Lagrangian methods solve (1.1) by a sequence of subproblems that minimize the augmented Lagrangian, either subject to a linearization of the constraints or as a bound-constrained problem.

4.1 ALGENCAN

Algorithmic Methodology: ALGENCAN is a solver based on an augmented Lagrangian-type algorithm (Andreani et al., 2007, 2008) in which a bound-constrained problem is solved in each iteration. The objective function in each iteration is the augmented Lagrangian of the original problem. This subproblem is solved by using a quasi-Newton method. The penalty on each constraint is increased if the previous iteration does not yield a better point.

Software and Technical Details: ALGENCAN is written in Fortran77, and interfaces are available for AMPL, C/C++, CUTEr, JAVA, MATLAB, Octave, Python, and R. The bound-constrained problem in each iteration is solved by using GENCAN, developed by the same authors. ALGENCAN and GENCAN are freely available under the GNU Public License from the TANGO project webpage (Martinez and Birgin, 2010).

4.2 GALAHAD

Algorithmic Methodology: GALAHAD (Gould et al., 2004b) contains a range of solvers for large-scale nonlinear optimization. It includes LANCELOT B, an augmented Lagrangian method with a nonmonotone descend condition; FILTRANE, a solver for feasibility problems based on a multi-dimensional filter (Gould et al., 2004a); and interior-point and active-set methods for solving large-scale quadratic programs (QPs). GALAHAD also contains a presolve routine for QPs (Gould and Toint, 2004) and other support routines.

Software and Technical Details: GALAHAD is a collection of thread-safe Fortran95 packages for large-scale nonlinear optimization. It is available in source form at (Gould et al., 2002). GALAHAD has links to CUTEr, and Fortran.

4.3 LANCELOT

Algorithmic Methodology: LANCELOT (Conn et al., 1992) is a large-scale implementation of a bound-constrained augmented Lagrangian method. It approximately solves a sequence of bound-constrained augmented Lagrangian problems by using a trust-region approach. Each trust-region subproblem is solved approximately: first, the solver identifies a Cauchy-point to ensure global convergence, and then it applies conjugate-gradient steps to accelerate the local convergence. LANCELOT provides options for a range of quasi-Newton Hessian approximations suitable for large-scale optimization by exploiting the group-partial separability of the problem functions.

Software and Technical Details: LANCELOT is written in standard ANSI Fortran77 and has been interfaced to CUTEr (Bongartz et al., 1995) and AMPL. The distribution includes installation scripts for a range of platforms in single and double precision. LANCELOT is available freely from Rutherford Appleton Laboratory, UK (Conn et al., 2010).

4.4 MINOS

Algorithmic Methodology: MINOS (Murtagh and Saunders, 1998) uses a projected augmented Lagrangian for solving general nonlinear problems. In each "major iteration" a linearly constrained nonlinear problem is solved where the linear constraints constitute all the linear constraints of (1.1) and also linearizations of nonlinear constraints. This problem is in turn solved iteratively by using a reduced-gradient algorithm along with a quasi-Newton algorithm. The quasi-Newton algorithm provides a search direction along which a line search is performed to improve the objective function and reduce the infeasibilities. MINOS does not guarantee convergence from a remote starting point. The user should therefore specify a starting point that is "close enough" for convergence. This issue will be addressed in a new software Knossos that will use a stabilized version of the linearly constrained Lagrangian method (Friedlander and Saunders, 2005). The user can also modify other parameters such as the penalty parameter in augmented Lagrangian to control the algorithm.

Software and Technical Details: MINOS is implemented in Fortran. The library can be called from Fortran and C programs and interfaces such as AMPL, GAMS, AIMMS, and MATLAB. The user must input routines for the function and gradient evaluations. If a gradient is not available, MINOS estimates it by using finite differences. MINOS is available under commercial and academic licenses from Stanford Business Software Inc.

4.5 PENNON

Algorithmic Methodology: PENNON (Kocvara and Stingl, 2003) is an augmented Lagrangian penalty-barrier method. In addition to standard NCOs, it can handle semidefinite NCOs that *include a semidefiniteness constraint on matrices of variables*. PENNON represents the semidefiniteness

constraint by computing an eigenvalue decomposition and adds a penalty-barrier for each eigenvalue to the augmented Lagrangian. It solves a sequence of unconstrained optimization problems in which the inequality constraints appear in barrier functions and the equality constraints in penalty functions. Every unconstrained minimization problem is solved with Newton's method. The solution of this problem provides a search direction along which a suitable merit function is minimized. Even though the algorithm has not been shown to converge for nonconvex problems, it has been reported to work well for several problems.

Software and Technical Details: PENNON can be called from either AMPL or MATLAB. The user manual (Kocvara and Stingl, 2008) provides instructions for input of matrices in sparse format when using AMPL. PENNON includes both sparse factorizations using the algorithm of Ng and Peyton (1993) and dense factorization using LAPACK. Commercial licenses and a free academic license are available from PENOPT-GbR.

5 Termination of NCO Solvers

Solvers for NCOs can terminate in a number of ways. Normal termination corresponds to a KKT point, but solvers can also detect (locally) infeasible NCOs and unboundedness. Some solvers also detect failure of constraint qualifications. Solvers also may terminate because of errors, and we provide some simple remedies.

5.1 Normal Termination at KKT Points

Normal termination corresponds to termination at an approximate KKT point, that is, a point at which the norm of the constraint violation, the norm of the first-order necessary conditions, and the norm of the complementary slackness condition, (**??**), are less than a user-specified tolerance. Some solvers divide the first-order conditions of (**??**) by the modulus of the largest multiplier. If the problem is convex (and feasible), then the solution corresponds to a global minimum of (1.1). Many NCO solvers include sufficient decrease conditions that make it less likely to converge to local maxima or saddle points if the problem is nonconvex.

5.2 Termination at Other Critical Points

Unlike linear programming solvers, NCO solvers do not guarantee global optimality for general nonconvex NCOs. Even deciding whether a problem is feasible or unbounded corresponds to a global optimization problem. Moreover, if a constraint qualification fails to hold, then solvers may converge to critical points that do not correspond to KKT points.

Fritz-John (FJ) Points. This class of points corresponds to first-order points at which a constraint qualification may fail to hold. NCO solvers adapt their termination criterion by dividing the stationarity condition by the modulus of the largest multiplier. This approach allows convergence

to FJ points that are not KKT points. We note that dividing the first-order error is equivalent to scaling the objective gradient by a constant $0 \leq \alpha \leq 1$. As the multipliers diverge to infinity, this constant converges to $\alpha \to 0$, giving rise to an FJ point.

Locally Inconsistent Solutions. To prove that an NCO is locally inconsistent requires the (local) solution of a feasibility problem, in which a norm of the nonlinear constraint residuals is minimized subject to the linear constraints; see (**??**). A KKT point of this problem provides a certificate that (1.1) is locally inconsistent. There are two approaches to obtaining this certificate. Filter methods switch to a feasibility restoration phase if either the stepsize becomes too small or the LP/QP subproblem becomes infeasible. Penalty function methods do not switch but drive the penalty parameter to infinity.

Unbounded Solutions. Unlike the case of linear programming where a ray along the direction of descent is necessary and sufficient to prove that the instance is unbounded, it is difficult to check whether a given nonlinear program is unbounded. Most NCO solvers use a user supplied lower bound on the objective and terminate if they detect a feasible point with a lower objective value than the lower bound.

5.3 Remedies If Things Go Wrong

If a solver stops at a point where the constraint qualifications appears to fail (indicated by large multipliers) or where the nonlinear constraints are locally inconsistent or at a local and not global minimum, then one can restart the solver procedure from a different initial point. Some solvers include automatic random restarts.

Another cause for failure is errors in the function, gradient, or Hessian evaluation (e.g., IEEE exceptions). Some solvers provide heuristics that backtrack if IEEE exceptions are encountered during the iterative process. In many cases, IEEE exceptions occur at the initial point, and backtracking cannot be applied. It is often possible to reformulate the nonlinear constraints to avoid IEEE exceptions. For example, $\log(x_1^3 + x_2)$ will cause IEEE exceptions if $x_1^3 + x_2 \leq 0$. Adding the constraint $x_1^3 + x_2 \geq 0$ does not remedy this problem, because nonlinear constraints may not be satisfied until the limit. A better remedy is to introduce a nonnegative slack variable, $s \geq 0$, and the constraint $s = x_1^3 + x_2$ and then replace $\log(x_1^3 + x_2)$ by $\log(s)$. Many NCO solvers will honor simple bounds (and interior-point solvers guarantee $s > 0$), so this formulation avoids some IEEE exceptions.

6 Calculating Derivatives

All solvers described above expect either the user or the modeling environment (AMPL, GAMS, etc.) to provide first-order and sometimes second-order derivatives. Some solvers can estimate first-order derivatives by finite differences. If the derivatives are not available or if their estimates

are not reliable, one can use several automatic differentiation (AD) tools that are freely available. We refer the readers to Griewank (2000) for principles and methods of AD and to Moré (2000) for using AD in nonlinear optimization. AD tools include ADOL-C (Griewank and Walther, 2004); ADIC, ADIFOR, and OpenAD (Hovland et al., 2009); and Tapenade (Hascoët and Pascual, 2004).

7 Web Resources

In addition to a range of NCO solvers that are available online, there exist an increasing number of web-based resources for optimization. The NEOS server for optimization (Czyzyk et al., 1998; NEOS, 2003) allows users to submit optimization problems through a web interface, using a range of modeling languages. It provides an easy way to try out different solvers. The NEOS wiki (Leyffer and Wright, 2008) provides links to optimization case studies and background on optimization solvers and problem classes. The COIN-OR project (COINOR, 2009) is a collection of open-source resources and tool for operations research, including nonlinear optimization tools (IPOPT and ADOL-C). A growing list of test problems for NCOs includes the CUTEr collection (Gould et al., 2010), Bob Vanderbei's collection of testproblems (Vanderbei, 2010b), the COPS testset (Dolan et al., 2010), and the GAMS model library (GAMS, 2010). The NLP-FAQ (Fourer, 2010) provides online answers to questions on nonlinear optimization.

Acknowledgments

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. This work was also supported by the U.S. Department of Energy through the grant DE-FG02-05ER25694.

References

Andersen, E. D., Jensen, B., Jensen, J., Sandvik, R., and Worsøe, U. (2009). Mosek version 6. Technical Report TR-2009-3, MOSEK.

Andreani, R., Birgin, E., Martinez, J., and Schuverdt, M. (2007). On augmented Lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18:1286–1309.

Andreani, R., Birgin, E., Martinez, J., and Schuverdt, M. (2008). Augmented Lagrangian methods under the constant positive linear dependence constraint qualification. *Mathematical Programming*, 111:5–32.

Benson, H., Sen, A., Shanno, D., and Vanderbei, R. (2006). Interior-point algorithms, penalty methods, and equilibrium constraints. *Computational Optimization and Applications*, 34(2):155–182.

Benson, H., Shanno, D., and Vanderbei, R. (2002). Interior-point methods for nonconvex nonlinear programming: filter-methods and merit functions. *Computational Optimization and Applications*, 23(2):257–272.

Bongartz, I., Conn, A. R., Gould, N. I. M., and Toint, P. L. (1995). CUTE: Constrained and Unconstrained Testing Enviroment. *ACM Transactions on Mathematical Software*, (21):123–160. `http://cuter.rl.ac.uk/cuter-www/interfaces.html`.

Bonnans, J., Gilbert, J., Lemaréchal, C., and Sagastizábal, C. (2006). *Numerical Optimization: Theoretical and Practical Aspects*. Springer, Berlin, 2nd edition.

Byrd, R. H., Gilbert, J. C., and Nocedal, J. (2000). A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89:149–185.

Byrd, R. H., Gould, N. I. M., Nocedal, J., and Waltz, R. A. (2004). An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. *Mathematical Programming, Series B*, 100(1):27–48.

Byrd, R. H., Hribar, M. E., and Nocedal, J. (1999). An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900.

Byrd, R. H., Nocedal, J., and Waltz, R. A. (2006). Knitro: An integrated package for nonlinear optimization. In di Pillo, G. and Roma, M., editors, *Large-Scale Nonlinear Optimization*, pages 35–59. Springer-Verlag.

COINOR (2009). COIN-OR: Computational infrastructure for operations research. `http://www.coin-or.org/`.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (1992). *LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A)*. Springer Verlag, Heidelberg.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (2010). LANCELOT webpage: `http://www.cse.scitech.ac.uk/nag/lancelot/lancelot.shtml`.

Czyzyk, J., Mesnier, M., and Moré, J. (1998). The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5:68–75. Try it at `www-neos.mcs.anl.gov/neos/`!

Dahl, J. and Vandenberghe, L. (2010). CVXOPT user's guide. Available online at `http://abel.ee.ucla.edu/cvxopt/userguide/`.

Dirkse, S. and Ferris, M. (1995). The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5:123–156.

Dirkse, S. and Ferris, M. (2010a). PATH-AMPL binaries: `ftp://ftp.cs.wisc.edu/math-prog/solvers/path/ampl/`.

Dirkse, S. and Ferris, M. (2010b). *PATHNLP*. GAMS. http://www.gams.com/dd/docs/solvers/pathnlp.pdf.

Dolan, E. D., Moré, J., and Munson, T. S. (2010). COPS large-scale optimization problems: http://www.mcs.anl.gov/~more/cops/.

Drud, A. (1985). A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31:153–191.

Drud, A. (2007). *CONOPT*. ARKI Consulting and Development, A/S, Bagsvaerd, Denmark. http://www.gams.com/dd/docs/solvers/conopt.pdf.

Ferris, M. and Munson, T. (1999). Interfaces to PATH 3.0: Design, implementation and usage. *Computational Optimization and Applications*, 12:207–227.

Fletcher, R. (1999). Stable reduced Hessian updates for indefinite quadratic programming. Numerical Analysis Report NA/187, University of Dundee, Department of Mathematics, Scotland, UK.

Fletcher, R. and de la Maza, E. S. (1989). Nonlinear programming and nonsmooth optimization by successive linear programming. *Mathematical Programming*, 43:235–256.

Fletcher, R. and Leyffer, S. (1998). User manual for filterSQP. Numerical Analysis Report NA/181, University of Dundee.

Fourer, R. (2010). Nonlinear programming frequently asked questions: http://wiki.mcs.anl.gov/NEOS/index.php/Nonlinear_Programming_FAQ.

Friedlander, M. P. and Saunders, M. A. (2005). A globally convergent linearly constrained lagrangian method for nonlinear optimization. *SIAM Journal on Optimization*, 15(3):863–897.

GAMS (2010). The GAMS model library index: http://www.gams.com/modlib/modlib.htm.

Gilbert, J. C. (2009). SQPlab A MATLAB software for solving nonlinear optimization problems and optimal control problems. Technical report, INRIA-Rocquencourt, BP 105, F-78153 Le Chesnay Cedex, France.

Gilbert, J. C. (2010). SQPlab website: http://www-rocq.inria.fr/~gilbert/modulopt/optimization-routines/sqplab/sqplab.html.

Gill, P., Murray, W., and Saunders, M. (2006a). User's guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming. Report, Dept. of Mathematics, University of California, San Diego.

Gill, P., Murray, W., and Saunders, M. (2006b). User's guide for SQOPT Version 7: Software for Large-Scale Nonlinear Programming. Report, Dept. of Mathematics, University of California, San Diego.

Gill, P., Murray, W., Saunders, M., and Wright, M. (1998). User's guide for NPSOL Version 5.0: A fortran package for nonlinear programming. Report SOL 86-1, Dept. of Mathematics, University of California, San Diego.

Gould, N., Orban, D., and Toint, P. (2010). CUTEr webpage: `http://cuter.rl.ac.uk/cuter-www/`.

Gould, N. I. M., Leyffer, S., and Toint, P. L. (2004a). A multidimensional filter algorithm for nonlinear equations and nonlinear least squares. *SIAM Journal on Optimization*, 15(1):17–38.

Gould, N. I. M., Orban, D., and Toint, P. L. (2002). *GALAHAD*. Rutherford Appleton Laboratory. `http://galahad.rl.ac.uk/`.

Gould, N. I. M., Orban, D., and Toint, P. L. (2004b). GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *ACM Trans. Math. Software*, 29(4):353–372.

Gould, N. I. M. and Toint, P. L. (2004). Presolving for quadratic programming. *Mathematical Programming*, 100(1):95132.

Griewank, A. (2000). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia.

Griewank, A., Juedes, D., Mitev, H., Utke, J., Vogel, O., and Walther, A. (1996). ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM TOMS*, 22(2):131–167.

Griewank, A. and Walther, A. (2004). ADOL-C a package for automatic differentiation of algorithms written in C/C++. `https://projects.coin-or.org/ADOL-C`.

Griewank, A., Walther, A., and Korzec, M. (2007). Maintaining factorized KKT systems subject to rank-one updates of Hessians and Jacobians. *Optimization Methods Software*, 22(2):279–295.

Hascoët, L. and Pascual, V. (2004). TAPENADE 2.1 user's guide. Technical Report 0300, INRIA.

Hovland, P., Lyons, A., Narayanan, S. H. K., Norris, B., Safro, I., Shin, J., and Utke, J. (2009). Automatic differentiation at Argonne. `http://wiki.mcs.anl.gov/autodiff/`.

Kocvara, M. and Stingl, M. (2003). PENNON—A code for convex nonlinear and semidefinite programming. *Optimization Methods and Software*, 18(3):317–333.

Kocvara, M. and Stingl, M. (2008). PENNON user's guide (version 0.9). Available at `http://www.penopt.com`.

Leyffer, S. and Wright, S. (2008). *NEOS wiki*. Argonne National Laboratory. http://wiki.mcs.anl.gov/NEOS/.

LINDO Systems Inc. (2010a). LINDO API 6.0 user manual.

LINDO Systems Inc. (2010b). Webpage. http://www.lindo.com/.

Martinez, J. and Birgin, E. (2010). TANGO: Trustable algorithms for nonlinear general optimization. Webpage http://www.ime.usp.br/~egbirgin/tango/index.php.

Moré, J. (2000). Automatic differentiation tools in optimization software. Technical Report ANL/MCS-P859-1100, Mathematics and Computer Science Division, Argonne National Laboratory.

Murtagh, B. and Saunders, M. (1998). MINOS 5.4 user's guide. Report SOL 83-20R, Department of Operations Research, Stanford University.

NEOS (2003). The NEOS server for optimization. Webpage, Argonne National Laboratory. http://www-neos.mcs.anl.gov/neos/.

Ng, E. and Peyton, B. (1993). Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM Journal on Scientific Computing*, 14(5):1034–1056.

Schenk, O., Waechter, A., and Hagemann, M. (2007). Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Journal of Computational Optimization and Applications*, 36(2-3):321–341. http://www.pardiso-project.org/.

Schittkowski, K. (1985). NLPQL: A Fortran subroutine for solving constrained nonlinear programming problems. *Annals of Operations Research*, 5(2):485–500.

Schittkowski, K. (2009). NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search – User's guide, Version 3.1. Report, Department of Computer Science, University of Bayreuth. http://www.math.uni-bayreuth.de/~kschittkowski/nlpqlp.htm.

Silva, R., Ulbrich, M., Ulbrich, S., and Vicente, L. N. (2010). Ipfilter homepage: http://www.mat.uc.pt/ipfilter/.

Ulbrich, M., Ulbrich, S., and Vicente, L. (2004). A globally convergent primal-dual interior-point filter method for nonconvex nonlinear programming. *Mathematical Programming*, 100:379–410.

Ulbrich, M., Ulbrich, S., and Vicente, L. N. (2008). A globally convergent primal-dual interior-point filter method for nonlinear programming: new filter optimality measures and computational results. Technical Report 08-49, Department of Mathematics, University of Coimbra.

Vanderbei, R. (2010a). LOQO homepage: `http://www.princeton.edu/~rvdb/loqo/LOQO.html`.

Vanderbei, R. (2010b). Nonlinear optimization models: `http://www.orfe.princeton.edu/~rvdb/ampl/nlmodels/`.

Vanderbei, R. and Shanno, D. (1999). An interior point algorithm for nonconvex nonlinear programming. *COAP*, 13:231–252.

Wächter, A. and Biegler, L. (2005a). Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization*, 16(1):32–48.

Wächter, A. and Biegler, L. (2005b). Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31.

Wächter, A. and Biegler, L. (2010). IPOPT project homepage: `https://projects.coin-or.org/Ipopt`.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Waltz, R. and Plantenga, T. (2009). KNITRO user's manual. Available online at `http://ziena.com/documentation.htm`.